

# Renewable energy production management with a new harmony search optimization toolkit

Ioannis Kougias<sup>1</sup> · Diamantis Karakatsanis<sup>2</sup> · Apostolos Malatras<sup>3</sup> ·  
Fabio Monforti-Ferrario<sup>1</sup> · Nicolaos Theodossiou<sup>2</sup>

Received: 5 February 2016 / Accepted: 24 March 2016 / Published online: 11 April 2016  
© The Author(s) 2016. This article is published with open access at Springerlink.com

**Abstract** The design goals of the developed software are described followed by its architecture, which is presented in detail. The GUI-based interface supports a variety of environmental management problems and can provide best practices in a timely manner. The toolkit is generic and applicable to any scientific field. It was applied on a renewable energy (RE) system's management. The developed model simulates the hydraulic characteristics of a small-scale hydropower (SHP) station. Harmony search algorithm (HSA) toolkit optimized the SHP's operation, without violating the ecological constraints related to environmental flow (EF) regimes. This was equal to maximizing the revenues from SHP's energy production in terms of a hypothetical fluctuating market. Apart from securing the provision of EF regimes, HSA toolkit's outcome provided management practices that increased the total economic gains. Supporting the economic viability of SHPs and their environmental friendliness is needed to strengthen their role in the RE mix.

**Keywords** Water resources management · Renewable energy · Energy management · Small hydropower · Optimization toolkit

## Introduction

During the second half of the twentieth century, meta-heuristics have attracted the interest of scientists and engineers dealing with complex optimization applications. Research concerning algorithmic design developed in parallel to the introduction of new optimization techniques. The majority of the developed algorithms were inspired from natural or artificial procedures in order to carry out their internal search process.

## Harmony search optimization algorithm

Music composition inspired (Geem et al. 2001) to introduce the harmony search algorithm (HSA). HSA imitates the way a musician plays within a music group, developing a selection process that optimizes specific problems. It belongs to the category of “neighborhood metaheuristics” that produce one possible solution per iteration. Initially, HSA was applied in urban water systems. Since then, there has been sustained and increasing interest in HSA applications, and apart from water engineering optimization problems (Kougias and Theodossiou 2013), HSA has been used on a vast variety of implementations (Manjarres et al. 2013).

Its efficiency has been extensively tested in the literature using established test functions and by comparing HSA's performance with other competitive techniques (Das et al. 2011). HSA's efficiency is also indicated by its popularity among those involved in the field of optimization and the rapid increase of the number of its applications.

## Optimization techniques

The application of metaheuristics can be a challenging task and often requires advanced programming skills, since

---

✉ Ioannis Kougias  
Ioannis.Kougias@ec.europa.eu

<sup>1</sup> Renewables & Energy Efficiency Unit, Institute for Energy & Transport, European Commission, Joint Research Centre, Via E. Fermi 2749, Ispra, Italy

<sup>2</sup> Department of Civil Engineering, Aristotle University of Thessaloniki, 541 24 Thessaloniki, Greece

<sup>3</sup> Digital Citizen Security Unit, Institute for the Protection and Security of the Citizen, European Commission, Joint Research Centre, Via E. Fermi 2749, Ispra, Italy

using metaheuristics normally involves developing ad hoc programs. Each program needs to correspond to the specifics of a single application and such an approach, apart from being time-consuming, imposes barriers to users with limited programming skills.

For this reason, a variety of optimization software in the form of user-friendly toolkits has been developed to answer the different needs. The development of such software started in the late twentieth century and ranges from programs developed in universities with an academic direction to commercial software developed in corporate environment.

### State of the art analysis

In “[Recent advances in optimization toolkit software](#)” section, the authors present an overview of existing optimization toolkits, their unique characteristics, the programming language used, and their strengths—limitations. A significant outcome of “[Recent advances in optimization toolkit software](#)” section’s survey was the absence of a generic HSA-based optimization software among the existing toolkits. Thus, users with limited computing skills are currently excluded from the use of HSA. At the same time in complex applications, where consecutive runs with different parameter-setting might be needed, HSA’s use is prohibitively labor-intensive.

The present research aims to cover this gap and provide an alternative, user-friendly approach. Accordingly, an optimization toolkit applicable to a wide extent of problems related to clean technologies has been developed. The toolkit’s novel characteristic is that in its core it uses harmony search, resulting to the first HSA toolkit built, so far. In “[The HSA optimization toolkit](#)” section, we present the design goals and the unique characteristics of the developed software, highlighting its contribution in bridging the existing gap in robust optimization tools that at the same time are easy-to-use.

## The HSA optimization toolkit

### Design goals

The main purpose of the HSA toolkit is to overcome the aforementioned barriers and provide an interface, where users input the objective function and the constraints in normal mathematical formula, in an easy, quick, and effective way. Thus, the main driving factor behind the development of our toolkit was to programmatically support the HSA by means of a GUI-based software framework, since to date there has been no similar effort.

Another design goal is to support a variety of problems, namely by allowing the unrestricted definition of variables

and constraints. This could be further supported by focusing the toolkit not only on a sole metaheuristic algorithm, but also offering a selection among variants. As mentioned in “[Recent advances in optimization toolkit software](#)” section, there is no ideal optimizer targeted at all types of problems. In this respect, the first version of our toolkit considered HSA and certain variations of HSA for optimization.

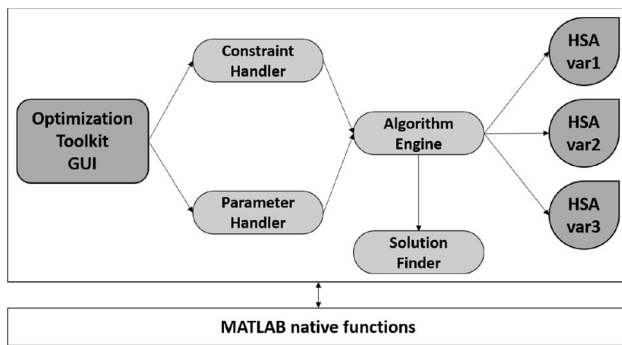
A third aim is to support the degree of responsiveness to decision-making. This is supported by the straightforward method of data input to the GUI that facilitates consecutive runs. Thus, updates and changes in the parameters of each application can be adopted in a timely manner, by altering the values of the corresponding equation constraint in the GUI. This feature is further described in “[Description of the HSA optimization toolkit](#)” section, showing the responsiveness of the developed toolkit, especially when compared to ad hoc designed software. Briefly, the design goals of the HSA optimization toolkit are

- (1) User-friendly, GUI-based optimization with HSA,
- (2) Support a variety of problems and applications,
- (3) Timely responsiveness.

### Description of the HSA optimization toolkit

The toolkit was built using MATLAB to benefit from its increased performance in dealing with advanced mathematical computations. MATLAB itself supports optimization through the optimization toolbox (MathWorks 2016). Moreover, many of the research works seen in “[Recent advances in optimization toolkit software](#)” section have also utilized the facilities that MATLAB has to offer to build optimization toolkits. Accordingly, we designed and developed a software environment to define and solve optimization problems using the HSA optimization algorithm. To the best of our knowledge, this is the first effort of kind to support the HSA. The toolkit allows for parameterization of the algorithm using an interactive graphical environment, for the definition of problem constraints and of the objective function, as well as the visualization of the results.

Figure 1 presents the architecture of the MATLAB-based HSA optimization toolkit. All functional components of the toolkit are built on and make use of MATLAB function calls for their operation. The optimization toolkit GUI is users’ entry point and the main component from which the optimization process commences. Subject to the users having entered the desired algorithmic parameters, the objective function, and problem constraints, there exist dedicated components to handle these variables. In particular, the constraint handler is in charge of parsing the constraints, while the parameter handler is responsible for



**Fig. 1** Architecture of the MATLAB-based HSA optimization toolkit

parsing the parameters. Both constraints and parameters are marshaled into data objects to be passed to the algorithm engine component of the toolkit. The latter component is fundamental in invoking the user-defined HSA variation and performing the actual optimization process according to the received constraints and parameters. Assuming successful completion of the optimization process, the solution finder component collects the list of results and displays it to the user.

The user interface of the toolkit is based on the established model-view-controller (MVC) paradigm (Krasner and Pope 1988), therefore separating the internal representation of information to the actual presentation of the information to the users. As illustrated in Fig. 2, the user is given the option to modify the values for the key

parameters of the HSA. Moreover, the user enters the definition for the objective function to be optimized, as well as the constraints to be taken into account during the optimization. The current implementation supports three variations of the HSA in regard to the handling of variables' values according to constraints, out of which the user needs to select the active variation to be applied for optimization. The three variations include

*HSA by choice*

In this case, both solutions stored in the memory and new candidate solutions are assigned values that strictly adhere to the constraints, thus averting the use of values that do not respect them.

*HSA by penalty*

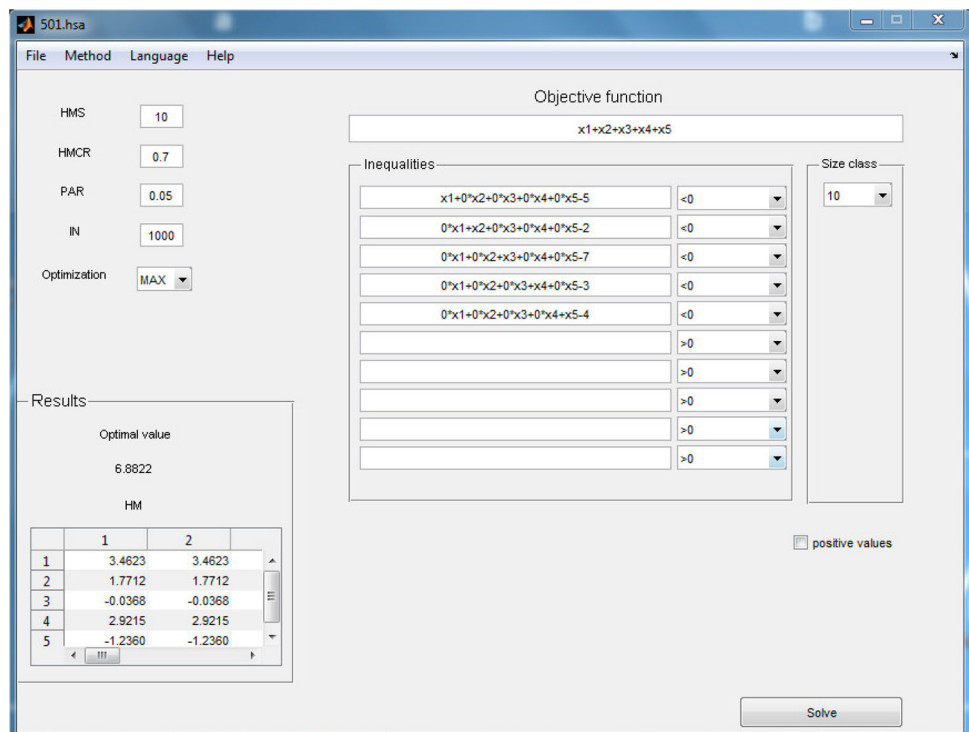
In this case, there is no restriction in considering values for the memory and every new harmony can assume values that negate the constraints. However, their evaluation as far as the objective function is concerned is subject to a penalty.

*HSA by range*

In this case, the constraints are completely ignored and the different variables are assigned random values from within a range of values arbitrarily defined by the user.

It should be noted that the toolkit also allows display and optionally store the results of an optimization process. The latest version of the HSA optimization toolkit is a

**Fig. 2** Main view of the graphical user interface of the HSA optimization toolkit



software of around 7000 lines of code and requires a MATLAB compiler runtime for its execution. Various optimization tests have been used to evaluate the toolkit's performance and the equations of the input boxes in Fig. 2 correspond to such a test. The toolkit's quite promising performance is also indicated by the execution time both in the test-function and in the case study of “[Application on renewable energy generation management](#)” section, for which the execution time was less than 1 min.

### Application on renewable energy generation management

The developed optimization toolkit has been applied in a real-world application, where the best management practices for a small-scale hydropower (SHP) station need to be identified. Since SHP's energy output is proportional to reservoirs' releases, water reserves are managed both seasonally and daily.

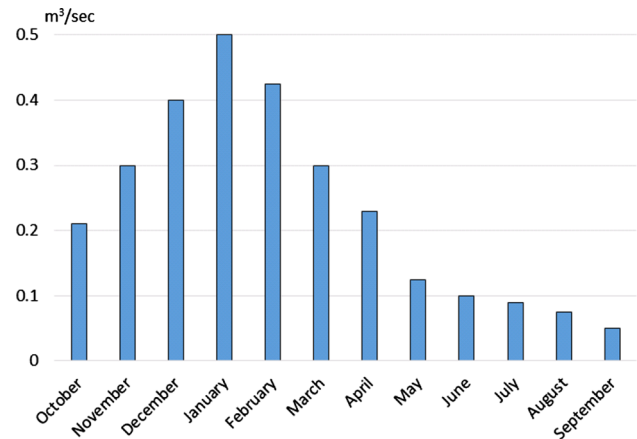
Seasonal management secures water availability in periods of low inflow, ensuring the reservoir will also serve a set of diverse functionalities (ecological conservation, energy production, irrigation-drinking water, flood mitigation, and recreation) on an intra-annual basis.

Daily management basically focuses on energy output, aiming to maximize the direct and indirect benefits of energy production, according to local conditions and policies. In the case of independent mini-grids or hybrid systems, the SHP's optimization aims to secure continuous energy supply, mini-grid balance, and minimum amounts of spilled energy (Moner-Girona et al. 2016). SHPs interconnected to a central grid sell the produced energy for a flat or varying price.

The present research analyzes the latter, SHPs connected to the main grid, selling electricity in prices proportional to demand. The developed model aims to increase the flexibility of SHPs' operation and fortify them against electricity price fluctuations. This is related to climate models' projections for significant variations on future water availability that might affect hydropower wholesale electricity prices and SHPs' economic viability.

### Characteristics of the small hydropower station

The studied SHP station has a design power capacity of 125 kW, with a crossflow-type turbine. The available hydraulic head is 20 m and the design maximum discharge  $Q_{\max} = 0.72 \text{ m}^3/\text{s}$ . The water flow of the rivulet flowing into the reservoir has been observed and recorded for several consecutive years, and Fig. 3 shows the monthly



**Fig. 3** Mean monthly water inflow rate to the reservoir under study

average of the recorded values that indicate expected monthly inflows.

The water discharge varies significantly throughout the year and becomes very low in summer. Naturally, the reservoir is gradually filled in the high-flow period (December–February) and contributes to the local needs in summer, when inflow is minimized (July–September). Considering that the active storage of the reservoir is  $140,000 \text{ m}^3$ , the average reservoirs' contribution during the low-flow period is estimated  $0.02 \text{ m}^3/\text{s}$ . Respectively, during the high-flow period water is stored with an equal rate (Table 1).

### Environmental flow

Among other considerations, water management needs to also concern environmental conservation. Environmental flow (EF) supports ecosystem sustainability, reflecting the

**Table 1** Monthly distribution of water availability ( $\text{m}^3/\text{s}$ )

Month	Avg. inflow	Env. flow	Stored water flow	Net flow
October	0.22	−0.07	0	0.15
November	0.30	−0.07	0	0.23
December	0.40	−0.07	−0.02	0.31
January	0.50	−0.07	−0.02	0.41
February	0.43	−0.07	−0.02	0.34
March	0.30	−0.07	0	0.23
April	0.22	−0.07	0	0.15
May	0.12	−0.07	0	0.05
June	0.10	−0.07	0	0.03
July	0.10	−0.07	+0.02	0.05
August	0.08	−0.07	+0.02	0.03
September	0.05	−0.07	+0.02	0

water volumes the ecosystem needs for its own functionalities. Thus, the facility maintains a minimum flow in the river, that is adequate for fish population, wildlife conservation, and water quality. In a recent study (Patsialis et al. 2014), the authors estimated the uniform EF value of the studied river, according to the current legislation:  $EF = 0.07 \text{ m}^3/\text{s}$ .

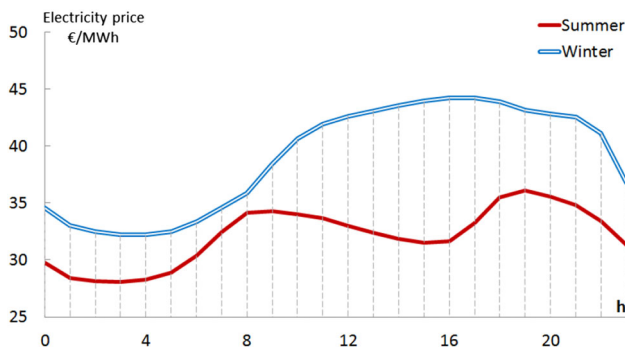
### Water availability for hydroelectric energy production

The minimum water discharge appears in September ( $Q^{\text{Sep}} = 0.05 \text{ m}^3/\text{s}$ ). Comparing September's discharge with the minimum flow, necessary for ecological conservation ( $EF = 0.07 \text{ m}^3/\text{s}$ ), it is evident that the available water inflow can not cover the ecological needs. Accordingly, in September, the entire rivulet flow and the contribution of stored water ( $Q = 0.02 \text{ m}^3/\text{s}$ ) needs to be used for ecological conservation. Such water shortages often force hydropower plants to curtail electricity production.

In Table 1, the average values of monthly flow-rate availability are presented. The second column corresponds to the average rivulet's inflow to the reservoir, while the third column illustrates the requirements for ecological conservation (EF regimes). The fourth column shows contributions to storage (negative values) or stored water releases (positive values). The fourth column present the net flow-rates for hydropower.

### Energy tariff structure

The present research shows the applicability and the advantages of the developed toolkit on the optimization of the daily operation of the SHP. It maximizes the economic benefits gained by energy production, utilizing the available water quantities. In this respect, a highly important factor is the energy pricing pattern, since production needs to be maximized in peak hours and minimized when energy demand and prices are low.



**Fig. 4** Hourly fluctuation of electricity prices in summer and winter

Energy prices are defined in the complex context of the energy market, whose mechanisms are beyond the scope of the present research. Attempting to simulate price fluctuations, a pricing pattern that varies both hourly and seasonally (Fig. 4) has been selected. Summer tariff is applied from May to September, corresponding to the different demand patterns between summer and winter. Naturally, the toolkit can adopt various pricing patterns or incentive schemes.

Water availability is identical in some months: October–April, November–March, May–July, and June–August (Table 1). Coincidentally, between these months, the same pricing policies apply; the first two pairs are subject to the winter tariff scheme and the remaining to the summer one. With water availability and pricing policies being identical, the management strategies in these months will be uniform.

## Results

### Model formulation

The energy production of a SHP is calculated from Eq. 1:

$$E_t = P_t \times T = (n_t \times \rho \times g \times Q_t \times H \times 1000) \times T, \quad (1)$$

where  $E_t$  is the produced electricity at hour  $t$  (kWh),  $P_t$  is the available power at hour  $t$  (kW),  $T$  is the duration of SHP's operation (h),  $n_t$  is the station's efficiency during hour  $t$ ,  $\rho$  is the water density ( $\text{kg}/\text{m}^3$ ),  $g$  is the gravitational acceleration ( $\text{m}/\text{s}^2$ ),  $Q_t$  is the average water flow during hour  $t$  ( $\text{m}^3/\text{s}$ ), and  $H$  is the hydraulic height (m).

### Objective function

The aim is to attain the maximum economic benefits, derived from selling the produced electricity to the grid. This is equal to maximizing the annual product of energy output times the energy price at each time step (Eq. 2):

$$\text{Benefit} = \sum_{m=1}^{12} \sum_{d=1}^{31} \sum_{t=1}^{24} E_t \times \text{Price}_t, \quad (2)$$

where  $E_t$  is the produced electricity at hour  $t$  (kWh),  $\text{Price}_t$  is the electricity price at hour  $t$  (EUR/kWh),  $m$  is month,  $d$  is number of days in each month (28–31).

The decision variables in Eq. 1 are the hourly values of discharge  $Q_t$ . Thus, the aim of the optimization is to define the  $Q_t$  values throughout the day, for all months of SHP's operation.

### Constraints

The optimization process is implemented for each month with the sum defining the objective function's value



(Eq. 2). This allows fulfilling the main constraint, i.e., serving all different water needs and respecting the EF regimes. Accordingly, for each month, the daily water availability constraint is (Eq. 3):

$$V_m^{\text{SHP}} \leq V_m^{\text{avail.}} = (Q_m^{\text{inflow}} - Q_m^{\text{EF}} \pm Q_m^{\text{stored}}) * 24 * 3600, \quad (3)$$

where  $V_m^{\text{SHP}}$  is the water volume directed to the SHP ( $\text{m}^3$ ),  $V_m^{\text{avail.}}$  is the water volume available for the SHP ( $\text{m}^3$ ),  $Q_m^i$  is the various water needs, as presented in Table 1 ( $\text{m}^3/\text{s}$ ).

Obviously, a second constraint defines all discharge values as positive (Eq. 4):

$$Q_t \geq 0. \quad (4)$$

### Results: optimization using HSA toolbox

The parameters for each month of operation have been successively inserted to the HSA toolkit. That included different water flow values, the energy tariff applicable to each period, and the constraints. Several runs have been implemented and the best management practices were detected.

The obtained best practices generally suggest small variations on the daily operation of the hydroelectric station. Hence the water discharge running the turbine will have slight hourly fluctuation, increasing the daily benefits. Between December and February, the best practice suggests an almost uniform flow. This is illustrated in Fig. 5 and can be justified by considering that during winter energy production is higher and the tariff scheme smoother.

On the contrary, in the period between April and October, the optimization suggested significant management changes throughout the day. This can be explained by the lower water availability in these months and subsequently SHP's lower energy production. The limited water

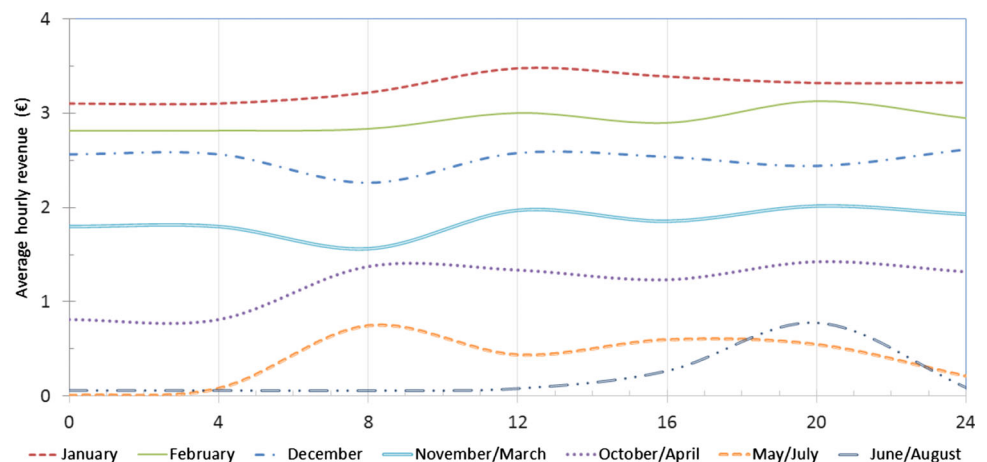
resources are not sufficient to fully operate the SHP, and tariff structure plays the major role in SHP's management. That explains the curves in Fig. 5 clearly corresponding to peak demand, maximizing energy production when energy prices reach their maximum.

The optimization increases the revenues, especially during low-flow months (+43.8 % in May, Table 2). Optimization's positive impact is significantly smaller in high-flow months, attenuating the overall annual gains.

### Recent advances in optimization toolkit software

There exists a great variety of optimization toolkits and frameworks aimed at supporting end users and researchers alike in conducting optimization experiments and analyzing their results. This variety is attributed to the plethora of available optimization algorithms, as well as to the diversity of the problems to which these algorithms are applied. In particular, different configuration parameters for each of the considered algorithms and a problem-specific scope that reflects lack of conformity in regard to the definition of constraints and goals, both motivate the large number of available optimization toolkits. As far as optimization is concerned, the seminal work presented in Wolpert and Macready (1997) validated that there is no optimal solution for all types of problems and that the performance of each algorithm greatly depends on the considered problem. Accordingly, for an optimization toolkit to be rendered broadly useful, a major functional requirement is its generality in terms of its support for various optimization algorithms. Additionally, other requirements include extensibility in supporting additional and possibly new algorithms and problems, portability in regard to its operational environment and usability in that it should be user-friendly (Houska et al. 2011).

**Fig. 5** Hourly variation of the electricity production revenue (best practice)



**Table 2** Monthly benefits with/without optimization of the operation

Months	Benefits (optimized)	Benefits (no optimization)	Increase (%)
January	1988.54	1979.96	+0.4
February	1467.20	1461.22	+0.7
March	1118.76	1110.71	+0.7
April	708.61	701.00	+1.1
May	209.03	199.31	+4.3
June	138.66	117.72	+19.8
July	209.03	199.30	+4.3
August	143.28	99.65	+43.8
September	0.00	0.00	–
October	732.23	724.38	+1.1
November	1082.67	1074.88	+0.7
December	1510.05	1497.04	+0.9

Early approaches in developing solutions to facilitate optimization processes mostly considered high-level programming interfaces (APIs) that would enable users to build their own optimization tools. Accordingly, *PSOt* (Birge 2003) is a MATLAB-based particle swarm optimization (PSO) toolkit that includes a suite of targeted custom function calls and low-level commands. The lack of flexibility and the narrow focus of this toolkit hinder its wide applicability, nevertheless other similar approaches were also proposed. Indicatively, *PSOTS* (Rui et al. 2009) is a SciLab toolbox with a Tcl/Tk interface that aims at single- and multi-objective optimization and in particular exploits PSO variants, *OPT++* (Meza et al. 2007) is a C++ library for nonlinear optimization, whereas the *ParadiseEO* (Liefoghe et al. 2007) software framework provides a generic C++ API for multi-objective meta-heuristic algorithms with the goal of supporting minimal programming on behalf of the users and a reusable design. Evidently, such approaches require a programming background and significant efforts from the users, while additionally aspects such as extensibility and generality are difficult to achieve using these interfaces.

Accordingly, works such as *pyOPT* (Perez et al. 2012) (Python-based object-oriented framework for nonlinear constrained optimization) and *PISA* (Bleuler et al. 2003) (generic interface for search algorithms) identified the previously mentioned limitations and proposed generic interfaces to support optimization processes independently of the underlying platforms and programming languages. Moreover, the considered decoupling of optimization constraints from application-specific formulations compensates to a great extent the lack of extensibility of such approaches. The work by Fang and Horstemeyer (2006) goes one step further in this direction by introducing a generic API for all multi-objective optimization problems with the goal of reducing the need for user programming.

While both the object-oriented approach undertaken and the measured performance (less than 8 % additional optimization time compared to hard-coded, optimizer-specific solutions) are promising, this work suffers from the fact that it does not actually achieve its goal and users need to be savvy in programming to perform optimization tasks.

To alleviate such concerns, GUI-based (graphical user interface) solutions were introduced in order to hide and abstract the complexity of optimization algorithms from the users and allow them to focus only on the functional behavior of these algorithms. The first generation of graphical optimization toolkits consisted of algorithm- or problem-specific solutions that lack generic nature. In this line of work, *TSPAntSim* (Aybars and Dogan 2009) is a web-based optimization software with a narrow focus on using ant colony optimization (ACO) as the optimizer and on solving problems that are based on the traveling salesman problem. While it is modular and extensible due to its object-oriented Java-based implementation, its scope is very limited to be widely used. Another object-oriented toolkit for optimization is *iOpt* (Voudouris et al. 2001), which separates presentation from program logic by making use of the MVC software pattern to present an interactive GUI that allows constraint-based problem modeling and optimization. Despite the fact that this work lacks experimental evaluation, it was one of the earliest developments of GUI-based optimization toolkits together with commercial systems such as *VisualDOC* (Balabanov et al. 2002), which provides a wealth of features, e.g., third party tool integration, database backend, variety of supported algorithms (genetic algorithms, evolutionary algorithms, PSO, and gradient-based optimization), albeit at a high cost and requiring excess computational resources. Further GUI-based optimization toolkits include *EVA2* (Kronfeld et al. 2010) (Java-based, client-server optimization framework for evolutionary metaheuristic algorithms),

*OPT4j* (Lukasiewicz et al. 2011) (modular metaheuristic optimization framework), *I-EMO* (Deb and Chaudhuri 2005) (optimization tool for genetic algorithms supporting their parameterization and results' visualization) and *HeuristicLab* (Wagner and Affenzeller 2005) (optimizer-independent and highly modular and extensible optimization environment supporting a great variety of evolutionary algorithms, such as ACO, PSO, and simulated annealing), while it should be noted that this listing is not meant to be exhaustive.

Recently, more advanced toolkits emerged that have extended functionality and address the majority of the aforementioned requirements. In this respect, *jMetal* (Durill and Nebro 2011) is a Java-based multi-objective optimization framework that supports a variety of evolutionary optimization algorithms and its object-oriented design and implementation inherently allow for code reusability, extensibility, and modularity. Moreover, it provides benchmark tests and quality indicators for performance assessment of various algorithms over a series of problems. A drawback of this framework is its lack of interactivity and the minimal support for decision-making, two aspects which are the focus of the toolbox presented in Tan et al. (2001). The latter involves users in the process of deciding on the optimal solution (also seen in the *HuGS* middleware platform Klau et al. 2002 that supports rapid prototyping of optimization systems) by means of a MATLAB-based GUI, but suffers from poor execution time compared to C/C++-based solutions and currently only supports a small number of evolutionary algorithms with limited extensibility. However, the fact that it allows for interactive, real-time optimization is a highly desirable feature. A similar approach is followed by the  $\mu GP$  evolutionary optimization toolkit (Sanchez et al. 2011) that is built using C++, but provides no GUI to end users and utilizes cumbersome XML configuration files to introduce problem constraints and algorithmic parameters. Its key feature is the fact that it involves end users in the optimization process by allowing them to set real-world constraints and thus guide the process towards the optimal solution. One of the most interesting and comprehensive open-source software environments for optimization is *LiGer* (Giagkiozis et al. 2013). It comprises a visual programming language that enables users to design their optimization problems as workflows, the execution of which yields the optimal solution. This Qt-based environment is quite extensible and has a focus on evolutionary optimization. Nonetheless, the latest prototype provides limited support for optimizers, namely differential evolution and PSO.

The use of optimization toolkits is prevalent in the clean energy production sector. A neural network toolbox has been developed to model and optimize the process of biogas recovery from landfills (Behera et al. 2015). As far as the energy efficiency field is concerned, simulation-

optimization toolkits have been applied to building performance analysis (Nguyen et al. 2014). MATLAB genetic algorithm toolbox has been recently applied on a multi-objective optimization problem, where an advanced steam power plant is optimized in terms both environmental and economic (Lira-Barragán et al. 2015). The same solver was used in a research on industrial waste optimal heat recovery (Gutiérrez-Arriaga et al. 2015). Such toolkits have also been applied in the optimization of off-grid electrification systems. The optimization of PV-diesel hybrid system for rural areas in Africa has been implemented with the use of the MATLAB optimization toolbox (Kusakana 2015).

Having examined the related literature as far as toolkits for optimization are concerned, we identified the major functional requirements to take into account in our proposed toolkit. In this respect, extensibility, generic nature both in terms of optimizers and problem definition, portability, efficiency, and usability are the key elements of our toolkit.

## Conclusions

The previous analysis shows HSA toolkit's successful use in a sustainable energy optimization problem. Its user-friendly environment, along with its performance and the graphical results, make it a valuable tool. Besides, the more complex the problems to be optimized, the more intense the necessity to adopt user-friendly, neat approaches.

Complex optimization applications are common in the energy field. The need for immediate, even real-time solutions is expected to raise as the share of intermittent energy sources grows. Complexity is also expected to increase as societies shift toward smart, interconnected electricity systems. SHPs role in the energy mix is to balance and mitigate the intermittency of PV and wind production. Thus, interventions to upgrade their power capacity (Kougias et al. 2016a), strategies to support their complementarity with other energy systems (Kougias et al. 2016b) and software supporting their flexible operation such as HSA toolkit can be proved necessary.

Future plans on the HSA toolkit include a version ported to Java, in order to benefit from its inherent portability and extensibility. The possibility to include further optimizers (e.g., ant colony and particle swarm) will promote flexibility and allow users to select the optimal one for their needs. This will also offer the ability to compare solutions obtained by different algorithms and locate global optimum solutions.

**Acknowledgments** The authors gratefully acknowledge use of the service of DG-JRC, supporting open access policy for scientific articles.



**Disclaimer** The views expressed in this paper are purely those of the writers and may not in any circumstances be regarded as stating an official position of the European Commission.

**Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

## References

- Aybars U, Dogan A (2009) An interactive simulation and analysis software for solving TSP using Ant Colony Optimization algorithms. *Adv Eng Softw* 40(5):341–349
- Balabanov V, Charpentier C, Ghosh D, Quinn G, Venter G (2002) Visualdoc: a software system for general purpose integration and design optimization. In: 9th AIAA/ISSMO symposium on multidisciplinary analysis and optimization
- Behera SK, Meher SK, Park H-S (2015) Artificial neural network model for predicting methane percentage in biogas recovered from a landfill upon injection of liquid organic waste. *Clean Technol Environ Policy* 17(2):443–453
- Birge B (2003) PSOT—a particle swarm optimization toolbox for use with MATLAB. In: Proceedings of the 2003 IEEE swarm intelligence symposium, 2003, SIS '03, p 182–186
- Bleuler S, Laumanns M, Thiele L, Zitzler E (2003). PISA: a platform and programming language independent interface for search algorithms. In: Proceedings of the 2nd international conference on evolutionary multi-criterion optimization, EMO'03. Springer, Berlin, p 494–508
- Das S, Mukhopadhyay A, Roy A, Abraham A, Panigrahi B (2011) Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization. *IEEE Trans Syst Man Cybern B* 41(1):89–106
- Deb K, Chaudhuri S (2005) I-EMO: an interactive evolutionary multi-objective optimization tool. In: Proceedings of the first international conference on pattern recognition and machine intelligence, PReMI'05. Springer, Berlin, p 690–695
- Durill J, Nebro A (2011) jMetal: a Java framework for multi-objective optimization. *Adv Eng Softw* 42(10):760–771
- Fang H, Horstemeyer M (2006) A generic optimizer interface for programming-free optimization systems. *Adv Eng Softw* 37(6):360–369
- Geem Z, Kim J, Loganathan G (2001) A new heuristic optimization algorithm: harmony search. *Simulation* 76(2):60–68
- Giagkiozis I, Lygoe R, Fleming P (2013) Liger: an open source integrated optimization environment. In: Proceedings of the 15th annual conference companion on genetic and evolutionary computation, GECCO, ACM, New York, NY, p 1089–1096
- Gutiérrez-Arriaga CG, Abdelhady F, Bamufleh HS, Serna-González M, El-Halwagi MM, Ponce-Ortega JM (2015) Industrial waste heat recovery and cogeneration involving organic Rankine cycles. *Clean Technol Environ Policy* 17(3):767–779
- Houska B, Ferreau H, Diehl M (2011) Acado toolkit. An open-source framework for automatic control and dynamic optimization. *Optim Control Appl Methods* 32(3):298–312
- Klau G, Lesh N, Marks J, Mitzenmacher M, Schafer G (2002) The hugs platform: a toolkit for interactive optimization. In: Proceedings of the working conference on advanced visual interfaces, AVI '02, ACM, New York, NY, p 324–330
- Kougias I, Theodossiou N (2013) Application of the harmony search optimization algorithm for the solution of the multiple dam system scheduling. *Optim Eng* 14:331–344
- Kougias I, Bódis K, Jäger-Waldau A, Monforti-Ferrario F, Szabó S (2016a) Exploiting existing dams for solar PV system installations. *Prog Photovolt Res Appl* 24(2):229–239
- Kougias I, Szabó S, Monforti-Ferrario F, Huld T, Bódis K (2016b) A methodology for optimization of the complementarity between small-hydropower plants and solar PV systems. *Renew Energy* 87(2):1023–1030
- Krasner G, Pope S (1988) A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *J Object Oriented Program* 1(3):26–49
- Kronfeld M, Planatscher H, Zell A (2010) The eva2 optimization framework. In: Proceedings of the 4th international conference on learning and intelligent optimization, LION'10. Springer, Berlin, p 247–250
- Kusakana K (2015) Operation cost minimization of photovoltaic-diesel-battery hybrid systems. *Energy* 85:645–653
- Liefvooghe A, Basseur M, Jourdan L, Talbi E (2007) Paradiseo-moeo: a framework for evolutionary multi-objective optimization. In: Proceedings of the 4th international conference on evolutionary multi-criterion optimization, EMO'07. Springer, Berlin, p 386–400
- Lira-Barragán LF, Gutiérrez-Arriaga CG, Bamufleh HS, Abdelhady F, Ponce-Ortega JM, Serna-González M, El-Halwagi MM (2015) Reduction of greenhouse gas emissions from steam power plants through optimal integration with algae and cogeneration systems. *Clean Technol Environ Policy* 17(8):2401–2415
- Lukasiewicz M, Glaß M, Reimann F, Teich J (2011) Opt4j: a modular framework for meta-heuristic optimization. In: Proceedings of the 13th annual conference companion on genetic and evolutionary computation, GECCO, ACM, New York, NY, p 1723–1730
- Manjarres D, Landa-Torres I, Gil-Lopez S, Ser JD, Bilbao M, Salcedo-Sanz S, Geem Z (2013) A survey on applications of the harmony search algorithm. *Eng Appl Artif Intell* 26(8):1818–1831
- MathWorks (2016) MATLAB optimization toolbox: user's guide, version R2016a. The MathWorks Inc., Natick, MA. Available at: [http://uk.mathworks.com/help/pdf\\_doc/optim/optim\\_tb.pdf](http://uk.mathworks.com/help/pdf_doc/optim/optim_tb.pdf)
- Meza J, Oliva R, Hough P, Williams P (2007) Opt++: an object-oriented toolkit for nonlinear optimization. *ACM Trans Math Softw* 33(2):12
- Moner-Girona M, Ghanadan R, Solano-Peralta M, Kougias I, Bódis K, Huld T, Szabó S (2016) Adaptation of feed-in tariff for remote mini-grids: Tanzania as an illustrative case. *Renew Sustain Energy Rev* 53:306–318
- Nguyen A-T, Reiter S, Rigo P (2014) A review on simulation-based optimization methods applied to building performance analysis. *Appl Energy* 113:1043–1058
- Patsialis T, Kougias I, Ganoulis J, Theodossiou N (2014) Irrigation dams for renewable energy production. In: Economics of water management in agriculture, vol 12. CRC Press, Taylor and Francis Group, Boca Raton, FL, p 270–294
- Perez R, Jansen P, Martins J (2012) pyopt: a Python-based object-oriented framework for nonlinear constrained optimization. *Struct Multidiscip Optim* 45(1):101–118
- Rui Q, Baogang H, Courmede P (2009) PSOTS: a particle swarm optimization toolbox in scilab. In: 2009 IEEE international workshop on open-source software for scientific computation (OSSC), p 107–114
- Sanchez E, Schillaci M, Squillero G (2011) Evolutionary optimization: the  $\mu$ GP toolkit. Springer, Boston
- Tan KC, Lee T, Khoo D, Khor EF (2001) A multiobjective evolutionary algorithm toolbox for computer-aided multiobjective optimization. *IEEE Trans Syst Man Cybern B* 31(4):537–556

- Voudouris C, Dorne R, Lesaint D, Liret A (2001) iopt: a software toolkit for heuristic search methods. In: Proceedings of the 7th international conference on principles and practice of constraint Programming, CP '01. Springer, London, p 716–719
- Wagner S, Affenzeller M (2005) Heuristiclab: a generic and extensible optimization environment. In: Ribeiro B, Albrecht R, Dobnikar A, Pearson D, Steele N (eds) Adaptive and natural computing algorithms. Springer, Vienna, pp 538–541
- Wolpert D, Macready W (1997) No free lunch theorems for optimization. *IEEE Trans Evol Comput* 1(1):67–82